

Name:

Student id:

Section: Serial#:

QUESTION #	1	2	3	4	TOTAL
MAX POINTS	20	20	20	24	
POINTS EARNED					

UNIVERSITY OF BAHRAIN

COLLEGE OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

TIME: 90 MINUTES

ITCS242: ASSEMBLY LANGUAGE PROGRAMMING

SECOND TEST

DATE: AUG 07, 2013

QUESTION ONE:

{20 pts}

- 1) Given: num dword ?; Write assembly instructions to perform: if left half of num is greater than the right half of num then num = lefthalf*righthalf, otherwise swap the two halves in num and then inverse odd-numbered bits in num.

```

MOV     AX, word ptr NUM + 2
CMP     AX, word ptr NUM
JA      L2
XCHG    AX, word ptr NUM          ;ROR  NUM, 16
MOV     word ptr NUM + 2, AX
XOR     NUM, 0AAAAAAAAH

- JMP    L3
L2: MUL  word ptr NUM
MOV     word ptr NUM, DX
SHL     NUM, 16
OR      WORD PTR NUM, AX
L3:

```

- 2) Write the needed instructions to store in AL and AH the counts of ONE and ZERO bits in a predefined memory location X of size double word. **Keep the value in X unchanged.**

```

MOV     AX, 0
MOV     ECX, 32
MOV     EBX, 0
L1: BT   X, EBX
JC      L2
INC     AH          ; ZEROS COUNTER
JMP     L3
L2: INC  AL          ; ONES COUNTER
L3: INC  EBX
LOOP    L1

```

Name:

Student id:

Section: Serial#:

QUESTION TWO: Write a sequence of assembly instructions to perform each of the following tasks:

- 1) Give ONE instruction to inverse the bits 4,7,10,12, and 14 in BX. Keep other bits unchanged. {2 pts}

```
XOR  BX, 5490H
```

- 2) Give no more than 3 instructions to calculate $ebx = ebx - eflags$. {3 pts}

```
PUSFD
POP   ECX
SUB   EBX, ECX
```

- 3) Give no more than 5 instructions to store in eax the product of multiplying the 2 halves in eax. {5 pts}

```
MOV    BX, AX
ROL    EAX, 16
IMUL   BX
SHL    EAX, 16
SHRD   EAX, EDX, 16
```

```
SHLD   EBX, EAX, 16
IMUL   BX
SHL    EAX, 16
SHRD   EAX, EDX, 16
```

- 4) Give no more than 6 instructions to divide the last 2 words pushed onto the stack and replace them by the quotient and remainder {5 pts}

```
POP    AX
POP    BX
CWD
IDIV   BX
PUSH   AX
PUSH   DX
```

- 5) Give no more than 5 instructions to shift the entire value in **EAX:EBX:ESI** 6 bits to the right. {5 pts}

```
MOV    ECX, 6
L2: SHR EAX, 1
      RCR EBX, 1
      RCR ESI, 1
      LOOP L2
```

```
SHRD   ESI, EBX, 6
SHRD   EBX, EAX, 6
SHR    EAX, 6
```

Name:

Student id:

Section: Serial#:

QUESTION THREE: What will be in the specified registers after executing the following code? { 20 pts}

a) MOV AX, 6C40H
MOV BX, 9E4FH
MUL BH

AX = 27 80 H

c) MOV AX, 7E9AH
TEST AX, 0BC37H
XOR AX, 0B7B7H

AX = 36 92 H

e) MOV AX, 3FFFH
MOV BX, 6750H
AND AX, BX
ROL AX, 8
RCR BX, 1

AX = 50 27 H

BX = B3 A8 H

g) MOV AX, 3FFFH
MOV BX, 5B70H
CMP AL, AH
JL L1
INC BL
JMP L2
L1: INC BH
L2:

BX = 5C 70 H

b) MOV AX, 2A3FH
MOV CX, 2090H
IDIV CL

AX = 3F A0 H

d) MOV AX, 2F08H
MOV BX, 794CH
SHRD AX, BX, 8

AX = 4C 2F H

f) MOV AX, 3FFFH
MOV BX, 6750H
MOV CL, 4
SAR AX, CL
NEG BX

AX = 03 FF H

BX = 98 B0 H

h) MOV AX, 3FFFH
MOV BX, 5B70H
CMP AL, AH
JB L3
DEC BL
JMP L4
L3: DEC BH
L4:

BX = 5B 6F H

Name:

Student id:

Section:

Serial#:

QUESTION FOUR:

Write each of the three parts in separate loop

{24 pts}

Write a complete Assembly program that defines an array BB consisting of 80 signed bytes, then:

- 1) Enters from the KBD 80 values in the range -128 to +127 and stores the entered values in array BB
- 2) Compares each two adjacent values in BB, places the larger value in array BG and the smaller value in array SM. You must define arrays BG and SM as required.
- 3) Stores in AH register the smallest value of array SM.

```
INCLUDE IRVINE32.INC

.DATA
BB sbyte 80 dup (?)
BG sbyte sizeof BB/2 dup (?)
SM sbyte sizeof BB/2 dup (?)

.CODE
; *****
START PROC
    LEA EBX, BB
    MOV ECX, LENGTHOF BB
LX: CALL READINT
    MOV [EBX], AL
    INC EBX
    LOOP LX
; *****
    LEA EBX, BB
    MOV ESI, 0
    MOV ECX, LENGTHOF BB/2
    MOV AH, [EBX]
    MOV AL, [EBX+1]
L1: CMP AH, AL
    JGE L2
    MOV SM[ESI], AL
    MOV BG[ESI], AH
    JMP L3
L2: MOV BG[ESI], AH
    MOV SM[ESI], AL
L3: ADD EBX, 2
    INC ESI
    LOOP L1
; *****
    MOV ECX, LENGTHOF SM
    MOV AH, SM[0]
    MOV EBX, 1
LX: CMP AH, SM[EBX]
    JL LY
    MOV AH, SM[EBX]
LY: INC EBX
    LOOP LX
    EXIT
START ENDP
END START
```